John von Neumann Institute for Computing

A Heterogeneous Pipelined Parallel Algorithm for Minimum Mean Squared Error Estimation with Ordered Successive Interference Cancellation

Francisco-Jose Martínez-Zaldívar,
Antonio M. Vidal-Maciá, Alberto González

published in

Parallel Computing: Architectures, Algorithms and Applications ,
C. Bischof, M. Bücker, P. Gibbon, G.R. Joubert, T. Lippert, B. Mohr, F. Peters (Eds.),
John von Neumann Institute for Computing, Jülich,
NIC Series, Vol. **38**, ISBN 978-3-9810843-4-4, pp. 263-270, 2007.
Reprinted in: Advances in Parallel Computing, Volume **15**,
ISSN 0927-5452, ISBN 978-1-58603-796-3 (IOS Press), 2008.

http://www.fz-juelich.de/nic-series/volume38

A Heterogeneous Pipelined Parallel Algorithm for Minimum Mean Squared Error Estimation with Ordered Successive Interference Cancellation

Francisco-Jose Martínez-Zaldívar[1], **Antonio. M. Vidal-Maciá**[2], **and Alberto González**[1]

[1] Departamento de Comunicaciones, Universidad Politécnica de Valencia
Camino de Vera s/n, 46022 Valencia, Spain
E-mail: {fjmartin, agonzal}@dcom.upv.es

[2] Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia
Camino de Vera s/n, 46022 Valencia, Spain
E-mail: avidal@dsic.upv.es

1 Introduction

Multiple Input Multiple Output (MIMO) systems have been extensively studied in recent years in the context of wireless communications. The original proposal by Foschini[1], known as BLAST (Bell Labs Layered Space-Time), has generated a family of architectures that uses multiple antenna arrays to transmit and receive information – with the aim of increasing the capacity and reliability of links. One of these architectures is V-BLAST (Vertical-BLAST). In this architecture, we can use linear decoders such as: Zero Forcing; MMSE; and the ordered version OSIC[2] (Ordered Successive Interference Cancellation), to be used in applications such as multicarrier systems[3] (i.e. OFDM —Orthogonal Frequency Division Multiplex— in Digital Video Broadcasting-Terrestrial or DVB-T) where the dimension of the problem may be around several thousands.

This paper describes a novel algorithm to solve the OSIC decoding problem and its parallelization — with better performance than those reported in the literature. Firstly, the basic OSIC decoding procedure is shown and cost comparisons made using two sequential algorithms. A pipelined parallelization of the sequential algorithm is derived: showing details of load balancing in homogeneous or heterogeneous networks; communications in message passing; shared memory architectures; and scalability. Finally, some experimental results are depicted.

2 OSIC Decoding Procedure

In a basic approach, it is necessary to solve the typical perturbed system $\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{v}$: where the known full rank matrix $\mathbf{H} = (\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_n) \in \mathbb{C}^{m \times n}$ represents the channel matrix; $\mathbf{y}$ is the observation vector; $\mathbf{x}$ is a vector whose components belong to a discrete symbol set; and $\mathbf{v}$ is process noise. The use of MMSE (Minimum Mean Square Error) estimation yields:

$$\hat{\mathbf{x}} = \left\lfloor \begin{pmatrix} \mathbf{H} \\ \sqrt{\alpha}\mathbf{I}_n \end{pmatrix}^{\dagger} \begin{pmatrix} \mathbf{y} \\ \mathbf{0} \end{pmatrix} \right\rceil = \lfloor \mathbf{H}_{\alpha}^{\dagger}\mathbf{y} \rceil \tag{2.1}$$

where $\lfloor \cdot \rceil$ denotes the mapping of the result in the symbol set, $\mathbf{H}_\alpha^\dagger$ (whose rows are named the *nulling vectors*) denotes the first m columns of the pseudoinverse of the *augmented* channel matrix $(\mathbf{H}^*, \sqrt{\alpha}\mathbf{I}_n)^*$, α^{-1} denotes a signal-to-noise ratio, and the asterisk superscript $(\cdot)^*$ denotes the complex conjugate. In OSIC, the signal components x_i, $i = 1, \ldots, n$, are decoded from the *strongest* (with the highest signal-to-noise ratio) to the *weakest*, cancelling out the contribution of the decoded signal component into the received signal and then repeating the process with the remaining signal components. Let $\mathbf{P}$ be the symmetric positive definite error estimation covariance matrix, $\mathbf{P} = \mathrm{E}\{(\mathbf{x} - \hat{\mathbf{x}})(\mathbf{x} - \hat{\mathbf{x}})^*\} = (\alpha\mathbf{I}_n + \mathbf{H}^*\mathbf{H})^{-1}$ that can be factorized as: $\mathbf{P} = \mathbf{P}^{1/2}\mathbf{P}^{*/2}$. These square roots factors are not unique, but it is advantageous if they have a triangular shape (i.e. Cholesky triangles). The index of the highest signal-to-noise ratio component of $\hat{\mathbf{x}}$ is the index of the lowest diagonal entry of $\mathbf{P}$: or the least Euclidean norm row index of $\mathbf{P}^{1/2}$. Let $(\mathbf{H}^*, \sqrt{\alpha}\mathbf{I}_n)^* = \mathbf{QL}$ be the QL factorization of the augmented channel matrix, where $\mathbf{L} \in \mathbb{C}^{n \times n}$ is a lower triangular matrix and the columns of $\mathbf{Q} \in \mathbb{C}^{(m+n) \times n}$ are orthogonal. Let us define $\mathbf{Q} = (\mathbf{Q}_\alpha^*, \mathbf{Q}_\beta^*)^*$, where $\mathbf{Q}_\alpha = (\mathbf{q}_{\alpha,1}, \mathbf{q}_{\alpha,2}, \ldots \mathbf{q}_{\alpha,n})$ are the first m rows of $\mathbf{Q}$. It is easy to verify that $\mathbf{L}^{-1} = \mathbf{P}^{1/2}$, and

$$\mathbf{H}_\alpha^\dagger = \mathbf{P}^{1/2}\mathbf{Q}_\alpha^*. \tag{2.2}$$

Let us suppose that $\mathbf{P}^{1/2}$ is a lower triangular matrix with $p_i^{1/2}, i = 1, \ldots, n$, as their diagonal entries and, for simplicity, with their rows sorted in increasing Euclidean norm order (otherwise, we obtain this using a permutation matrix $\mathbf{\Pi}$ and a unitary transformation $\mathbf{\Sigma}$ in $\mathbf{\Pi}\mathbf{H}_\alpha^\dagger = \mathbf{\Pi}\mathbf{P}^{1/2}\mathbf{\Sigma}\mathbf{\Sigma}^*\mathbf{Q}_\alpha^*$, while preserving the lower triangular structure in $\mathbf{\Pi}\mathbf{P}^{1/2}\mathbf{\Sigma}$). Using (2.2), the first *nulling vector* is $\mathbf{H}_{\alpha,1}^\dagger = p_1^{1/2}\mathbf{q}_{\alpha,1}^*$, and $\hat{x}_1 = \left\lfloor \mathbf{H}_{\alpha,1}^\dagger \mathbf{y} \right\rceil$. Now, in order to obtain $\hat{x}_2$ the process must be repeated with the *deflated* channel matrix $\mathbf{H}' = (\mathbf{h}_2, \mathbf{h}_3, \ldots, \mathbf{h}_n)$ and cancelling out the contribution of $\hat{x}_1$ in the received signal $\mathbf{y}' = \mathbf{y} - \hat{x}_1\mathbf{h}_1$. So, $\mathbf{P}'^{1/2}$ must be recomputed (to obtain the index of the least Euclidean norm of $\mathbf{P}'^{1/2}$, - again let us suppose that it is the first) and the QL factorization of the augmented *deflated* channel matrix. Hence, $\mathbf{H}_{\alpha,1}'^\dagger = p_1'^{1/2}\mathbf{q}_{\alpha,1}'^*$, and $\hat{x}_2 = \left\lfloor \mathbf{H}_{\alpha,1}'^\dagger \mathbf{y}' \right\rceil$.

With the previous assumptions, $\mathbf{P}'^{1/2}$ can be obtained directly[2] from the last $n-1$ rows and columns of $\mathbf{P}^{1/2}$, and $\mathbf{Q}_\alpha'$ from the last $n-1$ columns of $\mathbf{Q}_\alpha$, saving an order of magnitude computational cost. Therefore, only the $\mathbf{P}^{1/2}$ (matrix diagonal elements) and the $\mathbf{Q}_\alpha$ matrix are necessary to solve the OSIC problem. $\mathbf{P}^{1/2}$ and $\mathbf{Q}_\alpha$ or $\mathbf{H}_\alpha^\dagger$ can be computed solving a Recursive Least Squares (RLS) problem in a special way[2]. A subtle improvement in the execution time can be achieved[4] by obtaining the nulling vectors with $\mathbf{P}^{1/2}$ and $\mathbf{H}$. In both cases, the $\mathbf{P}^{1/2}$ matrix is needed. The algorithms will be developed for the ideas reported in Hassibi's paper[2]; because the results are extrapolable to the implementation proposed in Hufei Zhu's report[4]. To simplify the description of the algorithm, the details of the permutations and transformations due to the signal-to-noise ratio order will be omitted.

2.1 The Square Root Kalman Filter for OSIC

From (2.2), $\mathbf{Q}_\alpha = \mathbf{H}_\alpha^{\dagger *}\mathbf{P}^{-*/2}$. This matrix is propagated along the iterations of the square root Kalman Filter, which was initially devised to solve a Recursive Least Squares (RLS)

problem[2]. Below, a block version of the algorithm for OSIC[2] called SRKF-OSIC is reproduced.

Input: $\mathbf{H} = \left(\mathbf{H}_0^*, \mathbf{H}_1^*, \ldots, \mathbf{H}_{m/q-1}^*\right)^*$, $\mathbf{P}_{(0)}^{1/2} = \frac{1}{\sqrt{\alpha}}\mathbf{I}_n$ and $\mathbf{Q}_{\alpha,(0)} = \mathbf{0}$

Output: $\mathbf{Q}_\alpha = \mathbf{Q}_{\alpha,(m/q)}$, $\mathbf{P}^{1/2} = \mathbf{P}_{(m/q)}^{1/2}$

for $i = 0, \ldots, m/q - 1$ **do**

Calculate $\mathbf{\Theta}_{(i)}$ and applied in such a way that:

$$\mathbf{E}_{(i)}\mathbf{\Theta}_{(i)} = \begin{pmatrix} \mathbf{I}_q & \mathbf{H}_i\mathbf{P}_{(i)}^{1/2} \\ \mathbf{0} & \mathbf{P}_{(i)}^{1/2} \\ -\mathbf{\Gamma}_{(i+1)} & \mathbf{Q}_{\alpha,(i)} \end{pmatrix} \mathbf{\Theta}_{(i)} = \begin{pmatrix} \mathbf{R}_{e,(i)}^{1/2} & \mathbf{0} \\ \overline{\mathbf{K}}_{p,(i)} & \mathbf{P}_{(i+1)}^{1/2} \\ \mathbf{Z} & \mathbf{Q}_{\alpha,(i+1)} \end{pmatrix} = \mathbf{F}_{(i)}$$

end for

---

where $\mathbf{Z} = -\left(\mathbf{\Gamma}_{(i+1)}^* - \mathbf{H}_i\mathbf{H}_{\alpha,(i+1)}^\dagger\right)^* \mathbf{R}_{e,(i)}^{-*/2}$, q is the number of consecutive rows of $\mathbf{H}$ processed in a block (block size), so $\mathbf{H}_i \in \mathbb{C}^{q\times n}$. The iteration index subscript enclosed between parenthesis denotes that the variable is updated iteratively: $\mathbf{Q}_\alpha$ and $\mathbf{P}^{1/2}$ are the values $\mathbf{Q}_{\alpha,(i+1)}$ and $\mathbf{P}_{(i+1)}^{1/2}$ in the last iteration $i = m/q - 1$. $\mathbf{\Gamma}_{(i+1)} = \left(\mathbf{0}_{iq\times q}^T, \mathbf{I}_q, \mathbf{0}_{(m-q(i+1))\times q}^T\right)^T \in \mathbb{R}^{m\times q}$. $\mathbf{R}_{e,i}$ and $\overline{\mathbf{K}}_{p,(i)}$ are variables of the Kalman Filter whose meaning can be found in Sayed's paper[5], and $\mathbf{H}_{\alpha,(i+1)}^\dagger$ appears implicitly in $\mathbf{Z}$. The cost of one iteration is a matrix multiplication $\mathbf{H}_i\mathbf{P}_{(i)}^{1/2}$ and the application of a sequence of Givens rotations $\mathbf{\Theta}_{(i)}$, exploiting and preserving the triangular structure of $\mathbf{P}_{(i)}^{1/2}$ along the iterations. If we use a QR-like factorization algorithm we do not exploit the submatrix structure and the number of computations increase unnecessarily. Let $\mathrm{w}_{\text{TRMM}}(q,n)$ denote the cost of the $\mathbf{H}_i\mathbf{P}_{(i)}^{1/2}$ matrix multiplication (qn^2 flops[6]), where q and n are the dimensions of the result, and $\mathrm{w}_{\text{ROT}}(z)$, the cost of applying a Givens rotation to a pair of vectors of z components ($6z$ flops[6]). The cost can be approximated as:

$$\sum_{i=0}^{m/q-1}\left\{\mathrm{w}_{\text{TRMM}}(q,n) + \sum_{c=1}^{n}\sum_{r=1}^{q}\left[\mathrm{w}_{\text{ROT}}(q-r+1) + \mathrm{w}_{\text{ROT}}(n-c+1+[i+1]q)\right]\right\} =$$

$$= \sum_{i=1}^{m/q}\mathrm{w}_{\text{TRMM}}(q,n) + q\sum_{i=1}^{m/q}\sum_{c=1}^{n}\mathrm{w}_{\text{ROT}}(c+iq) + n\sum_{i=1}^{m/q}\sum_{r=1}^{q}\mathrm{w}_{\text{ROT}}(r) \approx 4n^2m + 3nm^2 \quad (2.3)$$

2.2 The Square Root Information Filter for OSIC

The *square root information filter* algorithm[5] variation to solve this problem, called SRIF-OSIC, is shown below. Let $\mathbf{V}_{(i)}^{(a)}$ and $\mathbf{W}_{(i)}^{(a)}$ be:

$$\mathbf{V}_{(i)}^{(a)} = \begin{pmatrix} \mathbf{P}_{(i)}^{-*/2} & \mathbf{H}_i^* \\ \mathbf{P}_{(i)}^{1/2} & \mathbf{0} \end{pmatrix}, \qquad \mathbf{W}_{(i)}^{(a)} = \begin{pmatrix} \mathbf{P}_{(i+1)}^{-*/2} & \mathbf{0} \\ \mathbf{P}_{(i+1)}^{1/2} & -\overline{\mathbf{K}}_{p,(i)} \end{pmatrix}$$

If $\mathbf{\Theta}_{(i)}$ is a unitary matrix such that $\mathbf{V}_{(i)}^{(a)}\mathbf{\Theta}_{(i)} = \mathbf{W}_{(i)}^{(a)}$, then $\mathbf{V}_{(i)}^{(a)}\mathbf{V}_{(i)}^{(a)*} = \mathbf{W}_{(i)}^{(a)}\mathbf{W}_{(i)}^{(a)*}$. Let the following augmented matrices be:

$$\mathbf{V}_{(i)}^{(b)} = \begin{pmatrix} \mathbf{P}_{(i)}^{-*/2} & \mathbf{H}_i^* \\ \mathbf{P}_{(i)}^{1/2} & \mathbf{0} \\ \mathbf{A} & \mathbf{B} \end{pmatrix}, \qquad \mathbf{W}_{(i)}^{(b)} = \begin{pmatrix} \mathbf{P}_{(i+1)}^{-*/2} & \mathbf{0} \\ \mathbf{P}_{(i+1)}^{1/2} & -\overline{\mathbf{K}}_{p,(i)} \\ \mathbf{L} & \mathbf{M} \end{pmatrix};$$

To propagate $\mathbf{Q}_{\alpha,(i)}$ along the iterations, let us force $\mathbf{A} = \mathbf{Q}_{\alpha,(i)}$ and $\mathbf{L} = \mathbf{Q}_{\alpha,(i+1)}$, and evaluate $\mathbf{V}_{(i)}^{(b)}\mathbf{V}_{(i)}^{(b)*} = \mathbf{W}_{(i)}^{(b)}\mathbf{W}_{(i)}^{(b)*}$. Hence, a solution can be obtained for $\mathbf{B} = \mathbf{\Gamma}_{(i+1)}$ and $\mathbf{M} = (\mathbf{\Gamma}_{(i+1)} - \mathbf{H}_{\alpha,(i)}^{\dagger *}\mathbf{H}_i^*)\mathbf{R}_{e,(i)}^{-*/2}$. The necessary ordering information can be obtained from $\mathbf{P}^{-*/2}$ without its (total) inversion, so avoiding the propagation of the second row of matrices in the algorithm. Accordingly, they will be deleted from $\mathbf{V}_{(i)}^{(b)}$ and $\mathbf{W}_{(i)}^{(b)}$ and the new matrices as will be denoted as $\mathbf{V}_{(i)}$ and $\mathbf{W}_{(i)}$.

Input: $\mathbf{H}^* = (\mathbf{H}_0^*, \mathbf{H}_1^*, \ldots, \mathbf{H}_{m/q-1}^*)$, $\mathbf{P}_{(0)}^{-1/2} = \sqrt{\alpha}\mathbf{I}$, $\mathbf{Q}_{\alpha,(0)} = \mathbf{0}$
Output: $\mathbf{Q}_\alpha = \mathbf{Q}_{\alpha,(m/q)}$, $\mathbf{P}^{-*/2} = \mathbf{P}_{(m/q)}^{-*/2}$
for $i = 0, \ldots, m/q - 1$ **do**
Compute $\mathbf{\Theta}_{(i)}$ in such a way that:

$$\mathbf{V}_{(i)}\mathbf{\Theta}_{(i)} = \begin{pmatrix} \mathbf{P}_{(i)}^{-*/2} & \mathbf{H}_i^* \\ \mathbf{Q}_{\alpha,(i)} & \mathbf{\Gamma}_{(i+1)} \end{pmatrix}\mathbf{\Theta}_{(i)} = \begin{pmatrix} \mathbf{P}_{(i+1)}^{-*/2} & \mathbf{0} \\ \mathbf{Q}_{\alpha,(i+1)} & \mathbf{M} \end{pmatrix} = \mathbf{W}_{(i)}$$

end for

Zeroes must be placed in the positions of the submatrix $\mathbf{H}_i^*$ in $\mathbf{V}_{(i)}$. This can be achieved by using Householder transformation applications or Givens rotation applications, or both, and right applied to $\mathbf{V}_{(i)}$. Let us suppose that Givens rotations are used, then for every row $r = 1, \ldots, n$ of $\mathbf{H}_i^*$, q Givens rotations need to be applied to a pair of vectors of $(n - r + 1) + [i + 1]q$ components, so the cost of the i^{th} iteration and the total iterations is:

$$W_{\text{sec},i} = \sum_{r=1}^{n} q\mathrm{w}_{\text{ROT}}((n - r + 1) + [i + 1]q) \; = \; 3qn^2 + 6q^2n[i + 1] + 3qn \quad (2.4)$$

$$W_{\text{sec}} = \sum_{i=0}^{m/q-1} W_{\text{sec},i} \; \approx \; 3n^2m + 3nm^2 \quad (2.5)$$

flops respectively. As a result, this version can be about 16% faster than the square root Kalman Filter based on (2.3) when $n \approx m$, because neither matrix multiplication, nor rotation applications, are necessary. This speedup could be even greater if Householder transformations are used (the higher the value of q, the greater the speedup, asymptotically to 25% when $n \approx m$).

3 Parallel Algorithm

For reasons of clarity, let us suppose as an example that we have $p = 2$ processors, P_0 and P_1. A matrix enclosed within square brackets with a processor subscript will denote

that part of the matrix belonging to such a processor. If it is enclosed within parenthesis, it denotes that the entire matrix is in such a processor. Let $\mathbf{C}_{(i)}$, $\mathbf{D}_{(i)}$ and $\mathbf{V}_{(i)}$ be:

$$\mathbf{C}_{(i)} = \begin{pmatrix} \mathbf{P}_{(i)}^{-*/2} \\ \mathbf{Q}_{\alpha,(i)} \end{pmatrix}, \ \mathbf{D}_{(i)} = \begin{pmatrix} \mathbf{H}_i^* \\ \mathbf{\Gamma}_{(i+1)} \end{pmatrix}, \ \mathbf{V}_{(i)} = \left(\mathbf{C}_{(i)}, \mathbf{D}_{(i)} \right)$$

The n columns of $\mathbf{C}_{(i)}$ will be distributed among the processors (n_0 columns belong to P_0 and n_1 columns to P_1, with $n_0 + n_1 = n$) and this assignment will not change during the parallel algorithm execution (it could change in an adaptive load balance algorithm version). $\mathbf{D}_{(i)}$ will be manipulated in a pipelined way by all the processors, so the subscript will change accordingly (it will be initially in P_0). We will divide $\mathbf{H}_i^*$ in $\left(\mathbf{D}_{(i)}\right)_{P_j}$ in p groups of n_j consecutive rows denoted by a left superscript. The initial data partition will be given by:

$$\mathbf{V}_{(i)} = \left(\begin{bmatrix} \mathbf{P}_{(i)}^{-*/2} \\ \mathbf{Q}_{\alpha,(i)} \end{bmatrix}_{P_0} \begin{bmatrix} \mathbf{P}_{(i)}^{-*/2} \\ \mathbf{Q}_{\alpha,(i)} \end{bmatrix}_{P_1} \begin{pmatrix} {}^{n_0}[\mathbf{H}_i^*] \\ {}^{n_1}[\mathbf{H}_i^*] \\ \mathbf{\Gamma}_{(i+1)} \end{pmatrix}_{P_0} \right) = \left(\left[\mathbf{C}_{(i)}\right]_{P_0} \left[\mathbf{C}_{(i)}\right]_{P_1} \left(\mathbf{D}_{(i)}\right)_{P_0} \right)$$

3.1 Processor Tasks

Let us suppose that P_0 gets zeroes in the n_0 rows of ${}^{n_0}[\mathbf{H}_i^*]$ by applying a sequence of unitary transformations $\mathbf{\Theta}_{(i),P_0}$ (the apostrophe $(\cdot)'$ will denote the updating of a matrix):

$$\mathbf{V}'_{(i)} = \mathbf{V}_{(i)}\mathbf{\Theta}_{(i),P_0} = \left(\begin{bmatrix} \mathbf{P}_{(i+1)}^{-*/2} \\ \mathbf{Q}_{\alpha,(i+1)} \end{bmatrix}_{P_0} \begin{bmatrix} \mathbf{P}_{(i)}^{-*/2} \\ \mathbf{Q}_{\alpha,(i)} \end{bmatrix}_{P_1} \begin{pmatrix} {}^{n_0}[\mathbf{0}] \\ {}^{n_1}[\mathbf{H}_i^*]' \\ \mathbf{\Gamma}'_{(i+1)} \end{pmatrix}_{P_0} \right)$$

It can be observed that the data not belonging to P_0 are not involved in the computations. $[\mathbf{P}_{(i)}^{-*/2}]_{P_0}$ is converted in $[\mathbf{P}_{(i+1)}^{-*/2}]_{P_0}$ and only the first $(i+1)q$ rows of the matrices $\mathbf{\Gamma}_{(i+1)}$ and $[\mathbf{Q}_{\alpha,(i)}]_{P_0}$ are updated due to the structure of $\mathbf{\Gamma}_{(i+1)}$ and the zero initial value of $\mathbf{Q}_{\alpha,(0)}$. It is also important to note that $\left[\mathbf{C}_{(i+1)}\right]_{P_0}$ (the first n_0 columns of the result $\mathbf{V}'_{(i)}$) are the first n_0 columns of the matrix $\mathbf{V}_{(i+1)}$. This is useful to obtain a pipelined behaviour in the processing work with minimum data movement from one iteration to the next. Now, if P_0 transfers ${}^{n_1}[\mathbf{H}_i^*]'$ and the nonzero part of $\mathbf{\Gamma}'_{(i+1)}$ to P_1, then P_1 can obtain zeroes in the n_1 rows of ${}^{n_1}[\mathbf{H}_i^*]'$ with the application of the unitary transformation sequence $\mathbf{\Theta}_{(i),P_1}$. Simultaneously, P_0 could work with $\left[\mathbf{C}_{(i+1)}\right]_{P_0}$ provided that new input data is available (pipelined behaviour):

$$\mathbf{V}''_{(i)} = \mathbf{V}'_{(i)}\mathbf{\Theta}_{(i),P_1} = \left(\begin{bmatrix} \mathbf{P}_{(i+1)}^{-*/2} \\ \mathbf{Q}_{\alpha,(i+1)} \end{bmatrix}_{P_0} \begin{bmatrix} \mathbf{P}_{(i+1)}^{-*/2} \\ \mathbf{Q}_{\alpha,(i+1)} \end{bmatrix}_{P_1} \begin{pmatrix} {}^{n_0}[\mathbf{0}] \\ {}^{n_1}[\mathbf{0}] \\ \mathbf{M} \end{pmatrix}_{P_1} \right) = \mathbf{W}_{(i)}$$

Then, the final result is obtained in a pipelined way by means of $\mathbf{V}_{(i)}\mathbf{\Theta}_{(i)} = \mathbf{V}_{(i)}\mathbf{\Theta}_{(i),P_0}\mathbf{\Theta}_{(i),P_1} = \mathbf{W}_{(i)}$. Let n_j be the number of columns of $\mathbf{C}_{(i)}$ assigned to P_j and r_{0_j}, the index of the first of them. It can be verified that the arithmetic overhead due to

parallelization is zero. The arithmetic cost in the P_j processor for the i^{th} iteration is:

$$W_{P_j,i} = \sum_{r=1}^{n_j} \text{w}_{\text{ROT}}(n - r_{0_j} + 2 - r + [i+1]q) = [6q(n - r_{0_j} + 2 + [i+1]q) - 3q]n_j - 3qn_j^2 \tag{3.1}$$

3.2 Load Balance, Heterogeneous Systems and Maximum Speedup

If the same number of columns of $\mathbf{C}_{(i)}$ are assigned to each processor, the parallel algorithm is clearly unbalanced due to to the lower triangular structure of $\mathbf{P}_{(i)}^{-*/2}$ in $\mathbf{C}_{(i)}$. The workload should be balanced at iteration level in order to minimize processor waiting time. The optimum number of columns n_j assigned to the processor P_j can be calculated solving the following second order equation:

$$W_{P_j,i} t_{w_j} = W_{\text{seq},i} t_{w_{\text{seq}}} / S_{\max}, \; \forall\, 0 \leq j \leq p-1 \tag{3.2}$$

beginning with n_0 and $r_{0_0} = 1$, up to n_{p-1}. $S_{\max}$ is the maximum speedup attainable in the heterogeneous or homogeneous system (in this case $S_{\max} = p$) and t_{w_j} and $t_{w_{\text{seq}}}$ are the time per flop in the P_j processor and the sequential processor, respectively. This result depends on the iteration index i, so if a static load balance scheme is desired then the load can balanced for the worst case, $i = m/q - 1$. An alternative way to balance the workload is to symmetrically assign columns to a processor, therefore the processor workload now depends linearly on the number of columns assigned to it, although the number of transfers nearly doubles. The maximum speedup in the heterogeneous network depends on the time per flop t_{w_j} of each processor. Let us define s_j as the normalized relative speed of the processor P_j (dimensionless): $s_j = \left(\sum_{r=0}^{p-1} \frac{tw_j}{t_{w_r}} \right)^{-1}$. It can be verified that $\sum_{j=0}^{p-1} s_j = 1$, and $t_{w_j} s_j = t_{w_k} s_k$, and if P_j is u times faster than P_k, then $s_j = u s_k$. Let us suppose that the sequential algorithm is run on the fastest processor P_f ($t_{w_{\text{seq}}} = t_{w_f}$). The maximum speedup can be obtained from (3.2) when a perfect load balance is obtained and there is no parallel arithmetic overhead. Hence, $S_{\max}$ can be obtained from (3.2) with $j = f$:

$$S_{\max} = \frac{W_{\text{seq},i} t_{w_f}}{W_{P_f,i} t_{w_f}} = \frac{\sum_{j=0}^{p-1} W_{P_j,i}}{W_{P_f,i}} = \frac{\sum_{j=0}^{p-1} W_{P_f,i} \frac{t_{w_f}}{t_{w_j}}}{W_{P_f,i}} = \frac{1}{s_f}$$

3.3 Communications, Shared Memory Implementation and Scalability

The parallel algorithm organization requires that processor P_j transfers the nonzero part of the updated $(\mathbf{D}_{(i)})_{P_j}$ matrix —$(i+1)q + \sum_{k=j+1}^{p-1} q n_k$ elements to P_{j+1}, $0 \leq j < p-1$. Let us suppose that the time for transferring this information from P_j to P_{j+1} for the i^{th} iteration can be modeled as a linear function of the number of elements to transfer. The worst case takes place when there is no computation and communication overlap, and all the transfers must be handled serially. For this worst case, the communication time is:

$$T_C = \sum_{i=0}^{m/q-1} \sum_{j=0}^{p-1} T_{C,P_j,i} = \mathbf{\Theta}\left(mnp\right) + \mathbf{\Theta}\left(mp^2\right) + \mathbf{\Theta}\left(\frac{pm^2}{q}\right)$$

If we use a shared memory multiprocessor system, and each process can be mapped on a processor. In this case, data transfer consists in copying this data from the memory space of P_j to the memory space of P_{j+1} and controlling access to it with a typical producer-consumer strategy. The copying of data to the next processor memory space can be a time consuming operation. We avoid this copying time by using a shared circular buffer whose elements are arrays of the same size as the data to transfer, so the symbolic copy can consist in a pointer to an element buffer update. These ideas can be extrapolated to shared and distributed memory parallel systems with minimum change in the code.

The scalability of the parallel system based on the isoefficiency function[7] can be evaluated by comparing the sequential time (2.5) with the total parallel overhead time. In our case, the only theoretical source of overhead is the communication time, so the total parallel overhead time is pT_C. If $W_{\text{sec}} t_{w_{\text{sec}}} = pT_C$, the workload must be increased as in the worst case of $n = \mathbf{\Theta}(p^2)$, $nm = \mathbf{\Theta}(p^3)$ or $\frac{n^2 q}{m} = \mathbf{\Theta}(p^2)$. If the transfers can be handled simultaneously, then $m, n = \mathbf{\Theta}(p)$.

3.4 Experimental Results

The tests have been run in a ccNUMA architecture multiprocessor running a 64 bit Linux operating system with up to 16 processors available to one user. Each processor is a 1.3 GHz Itanium 2. The programs have been coded in Fortran using a proprietary MPI communications library. Figure 1 shows the sequential execution time ratio of the SRKF-OSIC and SRIF-OSIC (Givens) versions for several values of q, where we can observe that the proposed SRIF-OSIC algorithm is faster than the SRKF-OSIC. We observed in the SRIF-OSIC algorithm sequential execution time an optimum value for q that gave a minimum execution time. There is no algorithmic reason for obtaining such minimum execution times, (2.5), so this behaviour is caused by the processor architecture. Figure 2 shows the efficiency of the proposed parallel algorithm in a message passing architecture for $q = 20$ and $m = 6000$, depicting a good efficiency in the results for $p = 2, 4, 8$ and 16 processors.

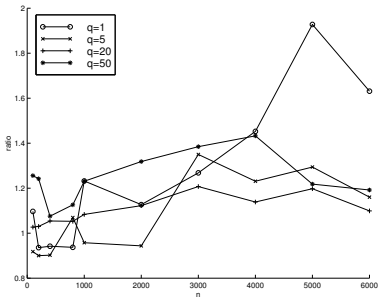

Figure 1. Execution time ratio of SRKF-OSIC and SRIF-OSIC (Givens) versions for $m = 6000$.

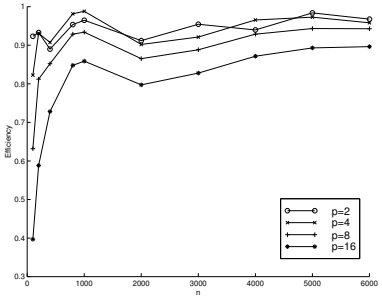

Figure 2. Efficiency of the SRIF-OSIC (Givens) parallel algorithm for $q = 20$ and $m = 6000$.

4 Conclusions

A novel algorithm is proposed to solve the OSIC decoding problem based on the square root information filter algorithm — and it offers a better performance than the reference based on the square root Kalman Filter algorithm. The improvement lies in the fact that a matrix multiplication, and the application of some rotations, are unnecessary in the new algorithm. A parallelization with a high degree of efficiency and scalability was obtained with a simple load balancing criterion for heterogeneous networks; and a simple extrapolation to shared memory, or distributed and shared memory systems.

Acknowledgements.

This work has been supported by the Spanish Government MEC and FEDER under grant TIC 2003-08238-C02.

References

1. G. J. Foschini, *Layered space-time architecture for wireless communications in a fading environment when using multiple antennas*, Bell Labs Tech. J., **1**, 41–59, (1996).
2. B. Hassibi, *An efficient square-root algorithm for BLAST*, IEEE International Conference on Acoustics, Speech and Signal Processing, **2**, II-737–II740, (2000).
3. Y.-S. Choi, P. J. Voltz and F. A. Cassara, *On channel estimation and detection for multicarrier signals in fast and selective Rayleigh fading channels*, IEEE Transactions on Communications, **49**, , (2001).
4. H. Zhu, Z. Lei and F. P. S. Chin, *An improved square-root algorithm for BLAST*, IEEE Signal Processing Letters, **11**, , (2004).
5. A. H. Sayed and Th. Kailath, *A state-space approach to adaptive RLS filtering*, IEEE Signal Processing Magazine, **11**, 18–60, (1994).
6. G. H. Golub and C. F. Van Loan, *Matrix Computations*, (Johns Hopkins University Press, Baltimore, 1996).
7. V. Kumar, A. Gram, A. Gupta and G. Karypis, *An Introduction to Parallel Computing: Design and Analysis of Algorithms*, (Addison-Wesley, 2003).